Sublinear Graph Algorithms and Randomized Numerical Linear Algebra

Michael W. Mahoney

ICSI and Dept of Statistics, UC Berkeley

(For more info, see:
http:// cs.stanford.edu/people/mmahoney/
or Google on "Michael Mahoney")

Motivation for StreamingNLA (1 of 2)

Data are medium-sized, but things we want to compute are "intractable," e.g., NP-hard or n^3 time, so develop an approximation algorithm.

- E.g., streaming for linear algebra on Spark/Hadoop/HPC

Data are large/Massive/BIG, so we can't even touch them all, so develop a sublinear approximation algorithm.

- E.g., fire hose style of streaming

Goal (in TCS streaming): Develop an algorithm s.t.:

Typical Theorem: My algorithm is faster than the exact algorithm, and it is only a little worse.

Motivation for StreamingNLA (2 of 2)

Mahoney, "Approximate computation and implicit regularization …" (PODS, 2012)

- Fact 1: I have *not* seen many examples (yet!?) where sublinear algorithms are a useful guide *for LARGE-scale "vector space" or "machine learning" analytics*

- Fact 2: I have seen real examples where sublinear algorithms are very useful, *even for rather small problems*, but their usefulness is *not* primarily due to the bounds of the Typical Theorem.

- Fact 3: I have seen examples where (both linear and sublinear) approximation algorithms yield "better" solutions than the output of the more expensive exact algorithm.

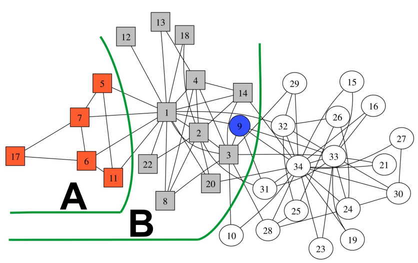
- *Sublinear/streaming algorithms involving matrices/graphs (read ML) are very different than other sublinear/streaming algorithms*

Anecdote 1: Communities in large informatics graphs

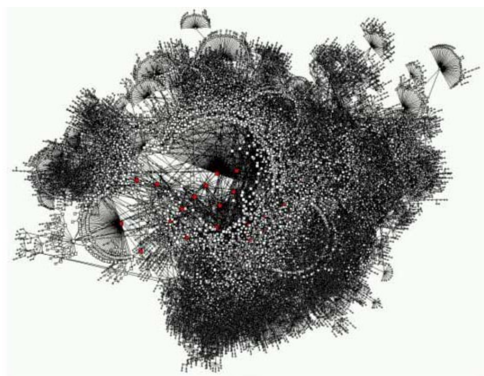
Mahoney "Algorithmic and Statistical Perspectives on Large-Scale Data Analysis" (2010)
Leskovec, Lang, Dasgupta, & Mahoney "Community Structure in Large Networks ..." (2009)

Data are expander-like at large size scales !!!

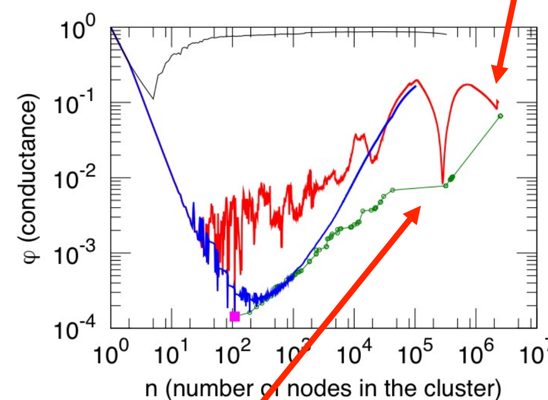
People imagine social networks to look like:



Real social networks actually look like:



Size-resolved conductance (degree-weighted expansion) plot looks like:



There do not exist good large clusters in these graphs !!!

How do we know this plot is "correct"?

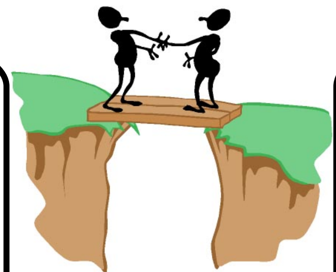
- (since computing conductance is intractable)
- Lower Bound Result; Structural Result; Modeling Result; Etc.
- Algorithmic Result (ensemble of sets returned by different approximation algorithms are very different)
- *Statistical Result* (Spectral provides more meaningful communities than flow)

Anecdote 2: Randomized Matrix Algorithms

Mahoney "Algorithmic and Statistical Perspectives on Large-Scale Data Analysis" (2010)
Mahoney "Randomized Algorithms for Matrices and Data" (2011)

Theoretical origins

- theoretical computer science, convex analysis, etc.
- Johnson-Lindenstrauss
- Additive-error algs
- Good worst-case analysis
- No statistical analysis
- No implementations



Practical applications

- NLA, ML, statistics, data analysis, genetics, etc
- Fast JL transform
- Relative-error algs
- Numerically-stable algs
- Good statistical properties
- *Beats LAPACK & parallel-distributed implementations on terabytes of data*

How to "bridge the gap"?

- decouple (implicitly or explicitly) randomization from linear algebra
- importance of statistical leverage scores!

The "core" RandNLA algorithm (1of2)

Drineas, Mahoney, etc., etc., etc. (200X, ...)

Problem: *Over-constrained* least squares (n x d matrix A,n >>d)

- Solve: $$\mathcal{Z} = \min_{x \in R^d} ||Ax - b||_2$$
- Solution: $$x_{opt} = A^{\dagger} b$$

Randomized Meta-Algorithm:

- For all i ε [n], compute *statistical leverage scores*: $$p_i = \frac{1}{d} ||U_{(i)}||_2^2$$
- Randomly sample O(d log(d)/ ε) rows/elements fro A/b, using *$\{p_i\}$ as importance sampling probabilities*.
- Solve the induced subproblem: $$\tilde{x}_{opt} = (SA)^{\dagger} Sb$$

Theorem: This gives 1±ε approximation, on the objective and the certificate (but you might fail and you have ε error and you are no faster).

The "core" RandNLA algorithm (2of2)

Drineas, Mahoney, etc., etc., etc. (200X, …)

A naïve implementation of this meta-algorithm might fail, has large ε error, and is no faster, but …

• Improve worst-case running time to O(nd log(d)) or O(nnz(A)+poly(d)) with smart random projections and/or smart leverage score approximation

• Use sketch as preconditioner of iterative algorithm and smart engineering to get O(log(1/ε)) to solve to machine precision and beat LAPACK w.r.t. wall-clock time

• Can solve least-squares and least absolute deviations on a terabyte of data to low/medium/high precision

• Implement in streaming environments by "grafting" this linear algebraic structure with projection sketches, heavy hitter sketches, etc.

• Can extend to get faster/more robust/more parallelizable low rank approximation of "nice" (e.g., PDE) and "not nice" (e.g., social media) data

• Can control statistical properties by worrying about small leverage scores and getting kernel-based methods with algorithmic/statistical bounds

Streaming/sublinear matrix/graph algorithms

Focus on linear algebraic or spectral graph structure

- Then graft onto more or less idealized streaming concepts
- This structure gives fast algorithmic and good statistical properties (but not always in the same way)

This is particularly necessary for “analyst in the loop” applications

- More relevant when you are “data knowledgeable” (science, national security, etc.)
- Less relevant when you are more data ignorant (e.g., internet search, social media, etc.)

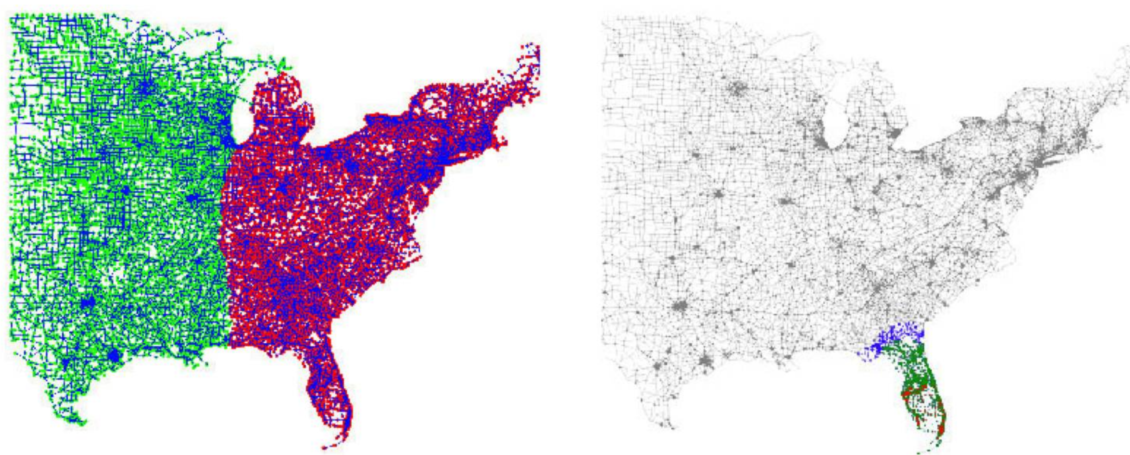
Local spectral optimization methods

Local spectral methods - provably-good local version of global spectral

ST04: truncated "local" random walks to compute locally-biased cut

ACL06: approximate locally-biased PageRank vector computations (with "push")

Chung08: approximate heat-kernel computation to get a vector



Q1: What do these procedures optimize approximately/exactly?

Q2: Can we write these procedures as optimization programs?

Recall spectral graph partitioning

The basic optimization problem:

$$\begin{aligned} \text{minimize} \quad & x^T L_G x \\ \text{s.t.} \quad & \langle x, x \rangle_D = 1 \\ & \langle x, 1 \rangle_D = 0 \end{aligned}$$

- Relaxation of:

$$\phi(G) = \min_{S \subset V} \frac{E(S, \bar{S})}{Vol(S)Vol(\bar{S})}$$

- Solvable via the eigenvalue problem:

$$\mathcal{L}_G y = \lambda_2(G) y$$

- Sweep cut of second eigenvector yields:

$$\lambda_2(G)/2 \leq \phi(G) \leq \sqrt{8\lambda_2(G)}$$

Also recall Mihail's sweep cut for a general test vector:

Thm.[Mihail] Let x be such that $< x, 1 >_D = 0$. Then there is a cut along x that satisfies $\frac{x^T L_G x}{x^T D x} \geq \phi^2(S)/8$.

Local spectral partitioning *ansatz*

Mahoney, Orecchia, and Vishnoi (2010)

Primal program:

$$\begin{array}{ll} \text{minimize} & x^T L_G x \\ \text{s.t.} & < x, x >_D = 1 \\ & < x, s >_D^2 \geq \kappa \end{array}$$

Interpretation:

- Find a cut well-correlated with the seed vector s.
- If s is a single node, this relax:

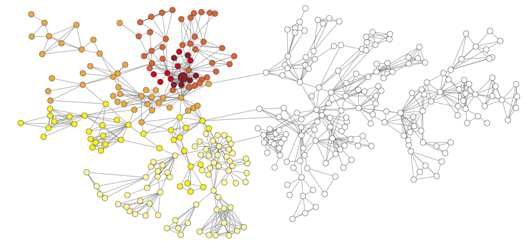
$$\min_{S \subset V, s \in S, |S| \leq 1/k} \frac{E(S, \bar{S})}{Vol(S) Vol(\bar{S})}$$

Dual program:

$$\begin{array}{ll} \max & \alpha - \beta(1-\kappa) \\ \text{s.t.} & L_G \succeq \alpha L_{K_n} - \beta \left(\frac{L_{K_T}}{\text{vol}(\bar{T})} + \frac{L_{K_{\bar{T}}}}{\text{vol}(T)} \right) \\ & \beta \geq 0 \end{array}$$

Interpretation:

- Embedding a combination of scaled complete graph K_n and complete graphs T and $\underline{T}$ (K_T and $K_{\underline{T}}$) - where the latter encourage cuts near (T,$\underline{T}$).



Main results

Mahoney, Orecchia, and Vishnoi (2010)

Algorithmic result, that computing the solution is "fast."

Theorem: If x^* is an optimal solution to LocalSpectral, it is a Generalized Personalized PageRank vector for parameter α, and it can be computed as solution to a set of linear eqns.

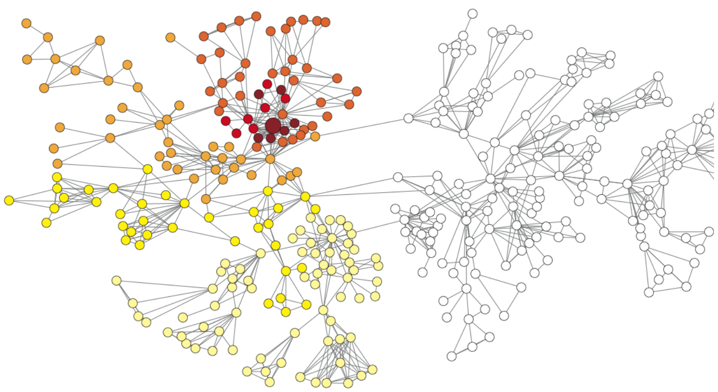
Upper bound, as usual from sweep cut & Cheeger.

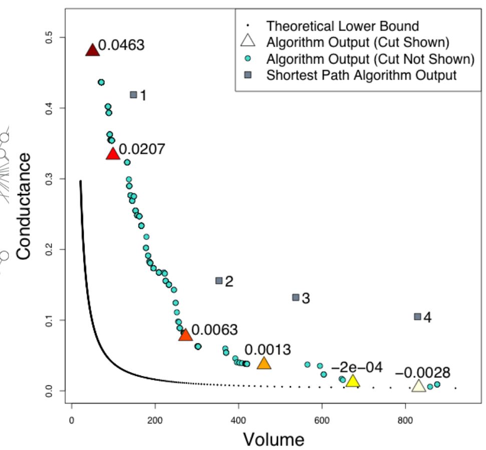
Theorem: If x^* is optimal solution to LocalSpect(G,s,κ), one can find a cut of conductance $\leq 8\lambda(G,s,\kappa)$ in time $O(n \lg n)$ with sweep cut of x^*.

Lower bound: Spectral version of flow-improvement algs.

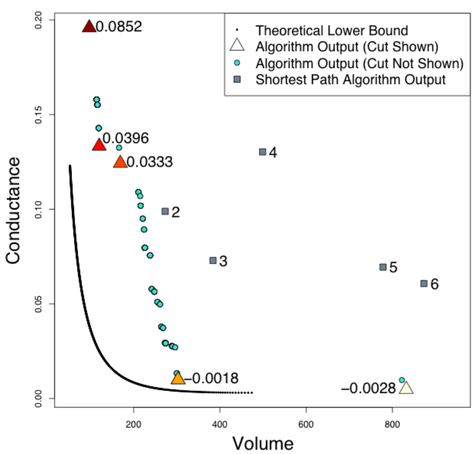
Theorem: Let s be seed vector and κ correlation parameter. For all sets of nodes T s.t. $\kappa' := \langle s, s_T \rangle_D^2$, we have: $\phi(T) \geq \lambda(G,s,\kappa)$ if $\kappa \leq \kappa'$, and $\phi(T) \geq (\kappa'/\kappa)\lambda(G,s,\kappa)$ if $\kappa' \leq \kappa$.

Illustration on small graphs



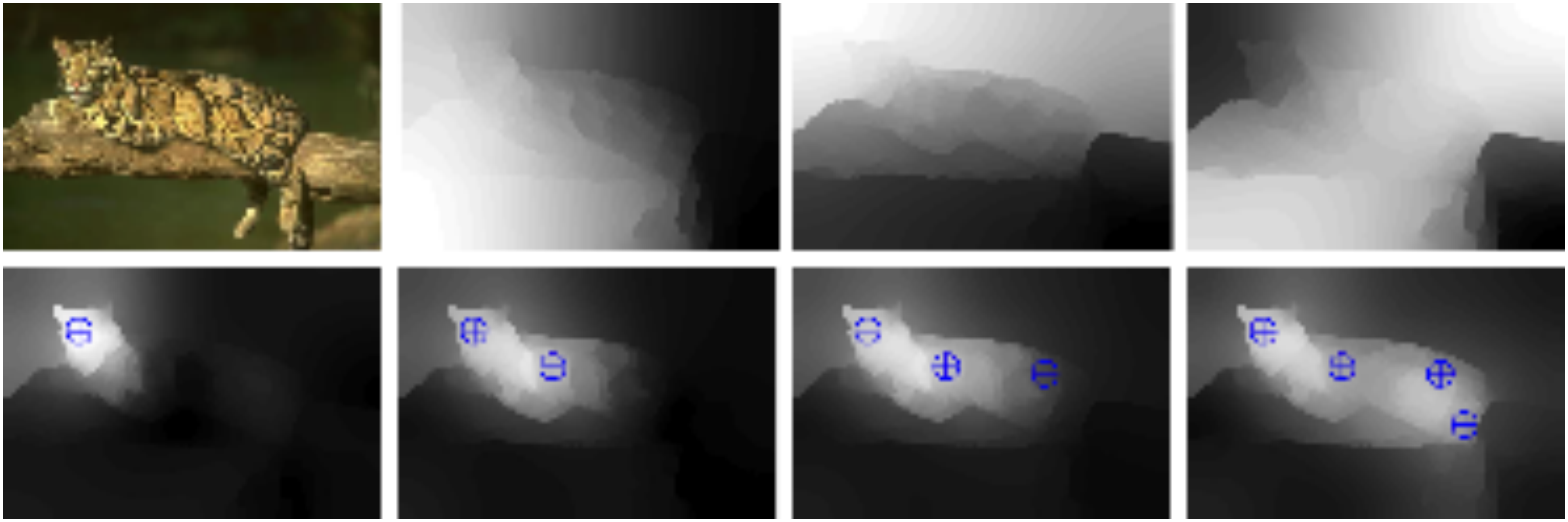


- Similar results if we do local random walks, truncated PageRank, and heat kernel diffusions.
- Often, it finds "worse" quality but "nicer" partitions than flow-improve methods. (Tradeoff we'll see later.)



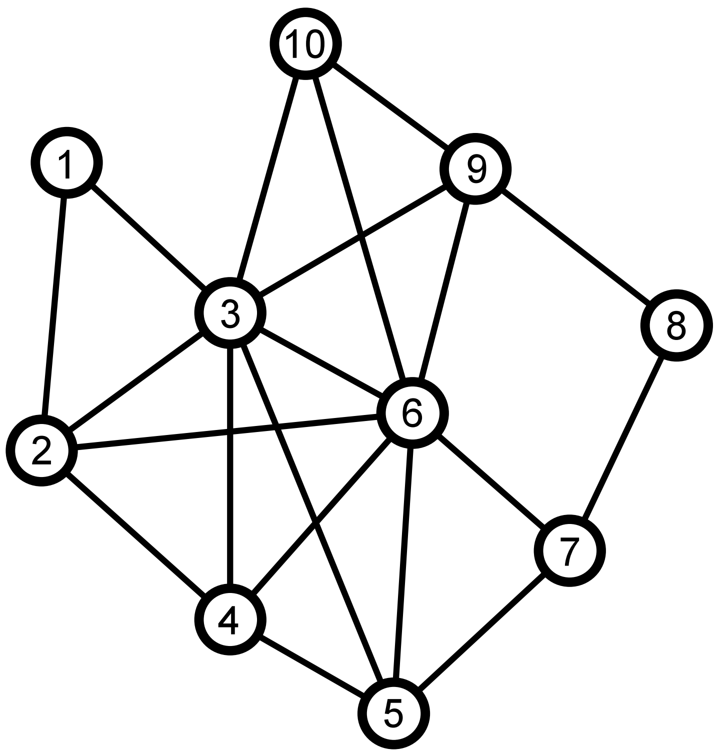
New methods are useful more generally

Maji, Vishnoi,and Malik (2011) applied Mahoney, Orecchia, and Vishnoi (2010)



- Cannot find the tiger with global eigenvectors.
- Can find the tiger with our LocalSpectral method!

Spectral algorithms and the PageRank problem/solution



- The PageRank random surfer
  1. With probability β, follow a random-walk step
  2. With probability (1-β), jump randomly ~ dist. **v**
- **Goal:** find the stationary dist. **x**

$$\mathbf{x} = \beta \mathbf{A}\mathbf{D}^{-1}\mathbf{x} + (1-\beta)\mathbf{v}$$

- **Alg:** Solve the linear system

$$(\mathbf{I} - \beta \mathbf{A}\mathbf{D}^{-1})\mathbf{x} = (1-\beta)\mathbf{v}$$

Symmetric adjacency matrix

Diagonal degree matrix

Solution

Jump vector

Push Algorithm for PageRank

- Proposed (in closest form) in Andersen, Chung, Lang (also by McSherry, Jeh & Widom) for *personalized PageRank*
  - Strongly related to Gauss-Seidel (see Gleich's talk at Simons for this)
- Derived to show improved runtime for balanced solvers

The Push Method τ, ρ

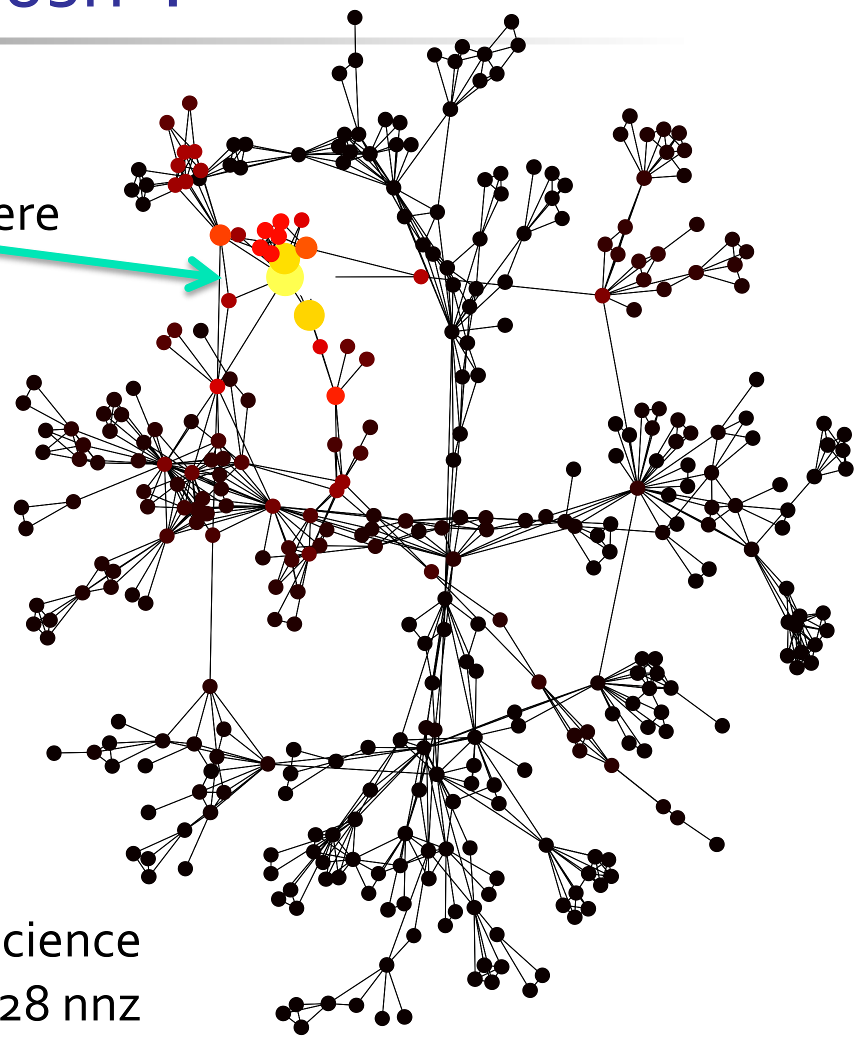
1. $\mathbf{x}^{(1)} = 0, \mathbf{r}^{(1)} = (1-\beta)\mathbf{e}_i, k = 1$
2. *while* any $r_j > \tau d_j$ (d_j *is the degree of node* j)
3. $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + (r_j - \tau d_j \rho)\mathbf{e}_j$
4. $$r_i^{(k+1)} = \begin{cases} \tau d_j \rho & i = j \\ r_i^{(k)} + \beta(r_j - \tau d_j \rho)/d_j & i \sim j \\ r_i^{(k)} & \text{otherwise} \end{cases}$$
5. $k \leftarrow k + 1$

Why do we care about "push"?

1. Used for empirical studies of "communities"
2. Used for "fast PageRank" approximation

- Produces *sparse* approximations to PageRank!
- Why does the "push method" have such empirical utility?

has a single one here



Newman's netscience
379 vertices, 1828 nnz
"zero" on most of the nodes

New connections between PageRank, spectral methods, localized flow, and sparsity inducing regularization terms

Gleich and Mahoney (2014)

- A new derivation of the PageRank vector for an undirected graph based on Laplacians, cuts, or flows
- A new understanding of the "push" methods to compute Personalized PageRank

- The "push" method is a sublinear algorithm with an implicit regularization characterization ...
- ...that "explains" it remarkable empirical success.

The s-t min-cut problem

Unweighted incidence matrix

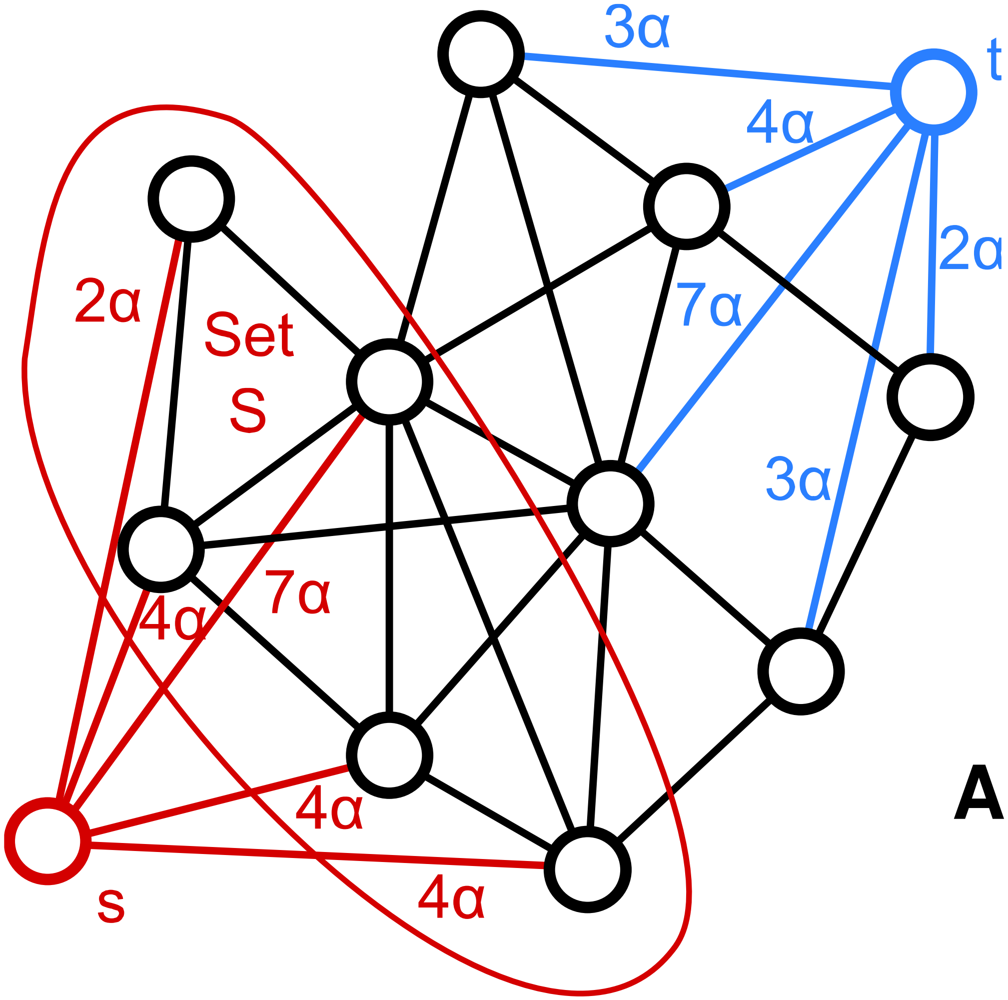
Diagonal capacity matrix

minimize $\quad \|\mathbf{Bx}\|_{C,1} = \sum_{ij \in E} C_{i,j} |x_i - x_j|$

subject to $\quad x_s = 1, x_t = 0, \mathbf{x} \geq 0.$

The localized cut graph

Gleich and Mahoney (2014)



Connect *s* to vertices in *S* with weight $\alpha \cdot$ degree

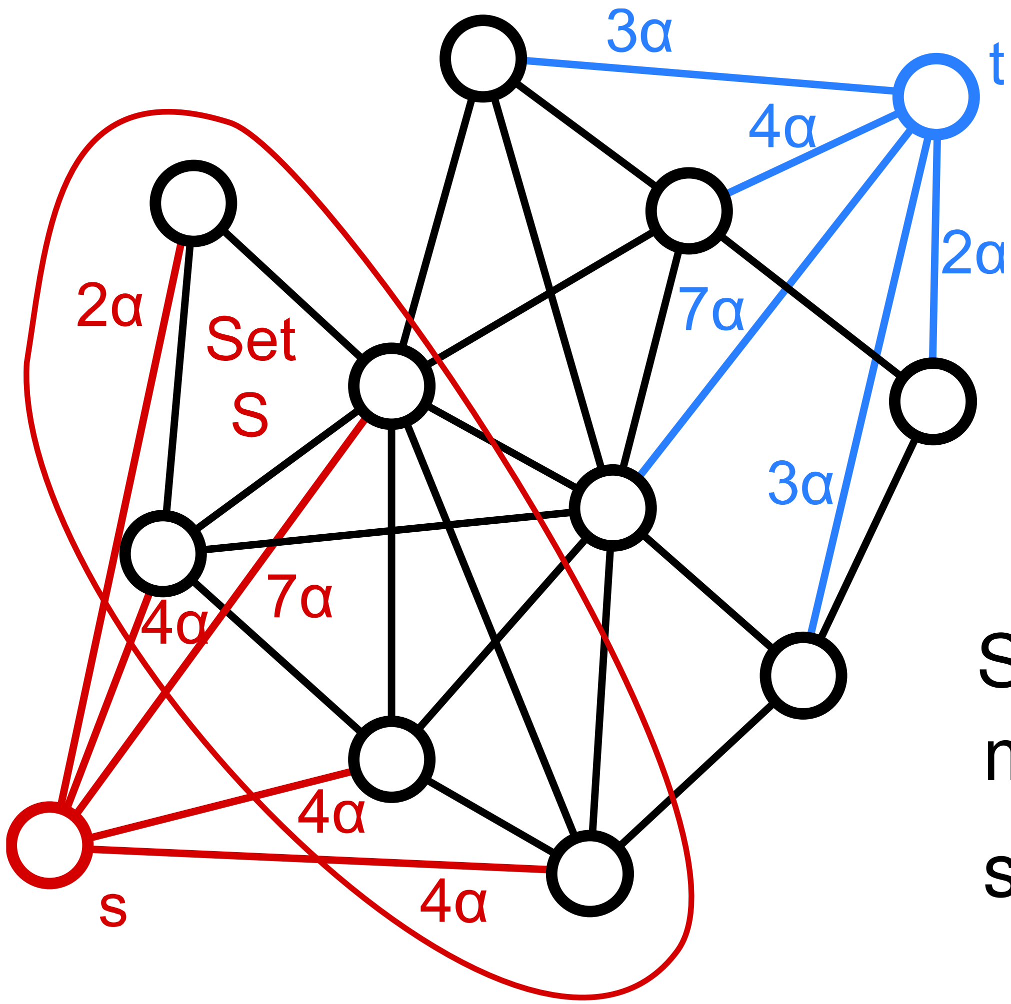
Connect *t* to vertices in $\bar{S}$ with weight $\alpha \cdot$ degree

- Related to a construction used in "FlowImprove" Andersen & Lang (2007); and Orecchia & Zhu (2014)

$$\mathbf{A}_S = \begin{bmatrix} 0 & \alpha \mathbf{d}_S^T & 0 \\ \alpha \mathbf{d}_S & \mathbf{A} & \alpha \mathbf{d}_{\bar{S}} \\ 0 & \alpha \mathbf{d}_{\bar{S}}^T & 0 \end{bmatrix}$$

The localized cut graph

Gleich and Mahoney (2014)



Connect *s* to vertices in *S* with weight $\alpha \cdot$ degree
Connect *t* to vertices in $\bar{S}$ with weight $\alpha \cdot$ degree

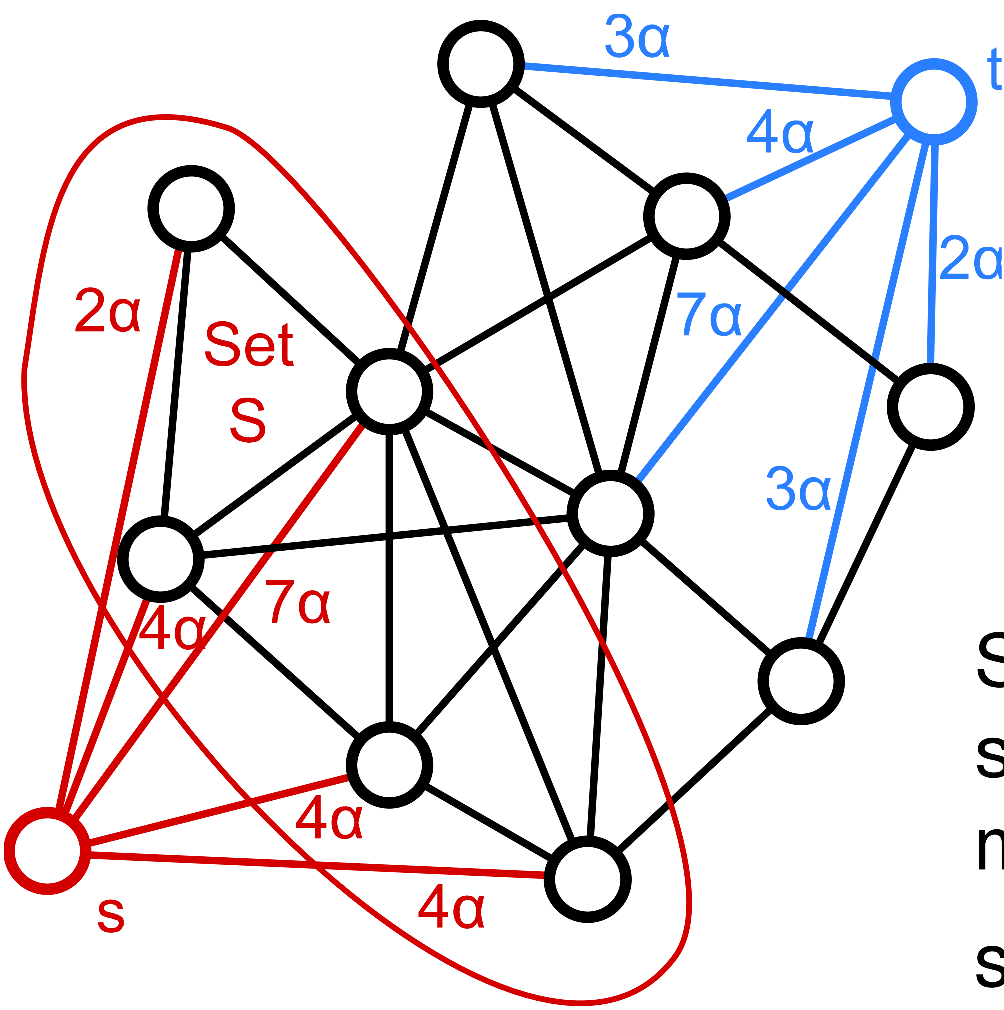
$$\mathbf{B}_S = \begin{bmatrix} \mathbf{e} & -\mathbf{I}_S & 0 \\ 0 & \mathbf{B} & 0 \\ 0 & -\mathbf{I}_{\bar{S}} & \mathbf{e} \end{bmatrix}$$

Solve the s-t min-cut

$$\begin{array}{ll} \text{minimize} & \|\mathbf{B}_S \mathbf{x}\|_{C(\alpha),1} \\ \text{subject to} & x_s = 1, x_t = 0 \\ & \mathbf{x} \geq 0. \end{array}$$

The localized cut graph

Gleich and Mahoney (2014)



Connect s to vertices in S with weight $\alpha \cdot$ degree
Connect t to vertices in $\bar{S}$ with weight $\alpha \cdot$ degree

$$\mathbf{B}_S = \begin{bmatrix} \mathbf{e} & -\mathbf{I}_S & 0 \\ 0 & \mathbf{B} & 0 \\ 0 & -\mathbf{I}_{\bar{S}} & \mathbf{e} \end{bmatrix}$$

Solve the "electrical flow" s-t min-cut

minimize $\|\mathbf{B}_S \mathbf{x}\|_{C(\alpha),2}$

subject to $x_s = 1, x_t = 0$

s-t min-cut -> PageRank

Gleich and Mahoney (2014)

The PageRank vector **z** that solves

$$(\alpha \mathbf{D} + \mathbf{L})\mathbf{z} = \alpha \mathbf{v}$$

with $\mathbf{v} = \mathbf{d}_S/\mathrm{vol}(S)$ is a renormalized solution of the electrical cut computation:

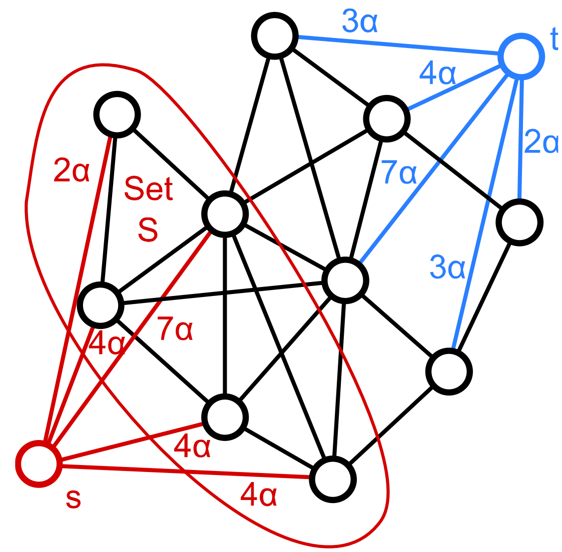
$$\begin{array}{ll} \text{minimize} & \|\mathbf{B}_S \mathbf{x}\|_{C(\alpha),2} \\ \text{subject to} & x_s = 1, x_t = 0. \end{array}$$

Specifically, if **x** is the solution, then

$$\mathbf{x} = \begin{bmatrix} 1 \\ \mathrm{vol}(S)\mathbf{z} \\ 0 \end{bmatrix}$$

Proof

Square and expand the objective into a Laplacian, then apply constraints.

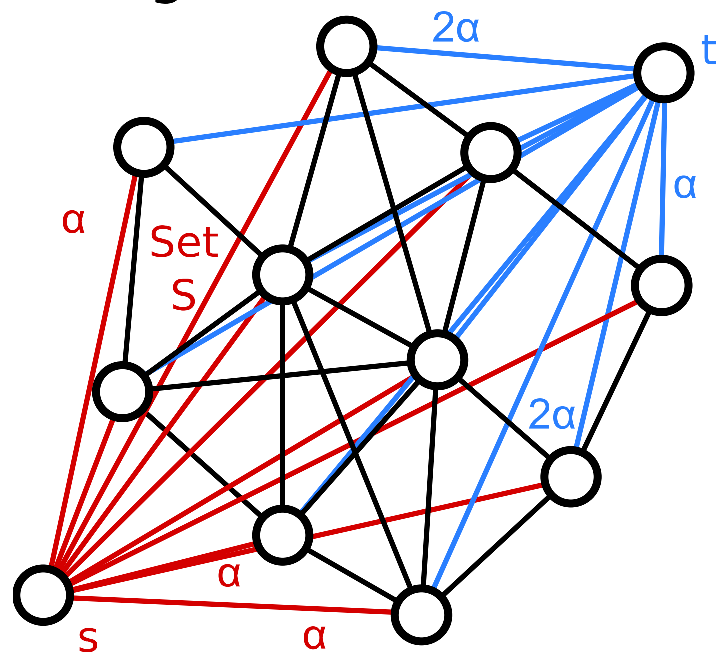


PageRank -> s-t min-cut

Gleich and Mahoney (2014)

- That equivalence works if **v** is degree-weighted.
- What if **v** is the uniform vector?

$$\mathbf{A}(\mathbf{s}) = \begin{bmatrix} 0 & \alpha \mathbf{s}^T & 0 \\ \alpha \mathbf{s} & \mathbf{A} & \alpha(\mathbf{d}-\mathbf{s}) \\ 0 & \alpha(\mathbf{d}-\mathbf{s})^T & 0 \end{bmatrix}.$$



- Easy to cook up popular diffusion-like problems and adapt them to this framework. E.g., semi-supervised learning (Zhou et al. (2004).

Back to the push method: sparsity-inducing regularization

Gleich and Mahoney (2014)

Let **x** be the output from the push method with $0 < \beta < 1$, $\quad \mathbf{v} = \mathbf{d}_S / \mathrm{vol}(S)$, $\rho = 1$, and $\tau > 0$.

Set $\alpha = \frac{1-\beta}{\beta}$, $\kappa = \tau \mathrm{vol}(S)/\beta$, and let $\mathbf{z}_G$ solve:

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}\|\mathbf{B}_S \mathbf{z}\|_{C(\alpha),2}^2 + \kappa \|\mathbf{D}\mathbf{z}\|_1 \\ \text{subject to} & z_s = 1, z_t = 0, \mathbf{z} \geq 0 \end{array},$$

Need for normalization

Regularization for sparsity

where $\mathbf{z} = \begin{bmatrix} 1 \\ \mathbf{z}_G \\ 0 \end{bmatrix}$.

Then $\mathbf{x} = \mathbf{D}\mathbf{z}_G / \mathrm{vol}(S)$.

Proof Write out KKT conditions
Show that the push method solves them. Slackness was "tricky"

Success strategy for RandNLA

“Decouple” randomness from vector space structure

Importance of statistical leverage scores (a “non-pathological” problem-specific complexity measure)

This led to:

- Much better worst-case bounds (in theoretical computer science)
- Much better statistical properties (in machine learning and statistics)
- Much better implementations (in RAM, parallel, distributed, etc.)
- Much better usefulness in applications (genetics, astronomy, imaging, etc.)

Success strategy for Sublinear/Streaming Graph (and Matrix, i.e., ML) Analytics

Don't over-optimize to worst-case analysis

• matrices (including spectral graph theory) are much more structured objects than general metric spaces

• so the bar is higher to get fine results (think all of NLA and scientific computing)

Need more realistic models of data presentation (details of data presentation/layout matter a lot)

• often a tradeoff between speed and statistical meaningfulness

Understand implicit statistical properties in scalable algorithms

• this gives "better" algorithms for even modest-sized data